

RRRRRRRR	MM	MM	11	RRRRRRRR	EEEEEEEEEE	LL	BBBBBBBB	LL	KK	KK	
RRRRRRRR	MM	MM	11	RRRRRRRR	EEEEEEEEEE	LL	BBBBBBBB	LL	KK	KK	
RR	RR	MMM	1111	RR	RR	LL	BB	LL	KK	KK	
RR	RR	MMM	1111	RR	RR	LL	BB	LL	KK	KK	
RR	RR	MM	11	RR	RR	LL	BB	LL	KK	KK	
RR	RR	MM	11	RR	RR	LL	BB	LL	KK	KK	
RRRRRRRR	MM	MM	11	RRRRRRRR	EEEEEEEEEE	LL	BBBBBBBB	LL	KKKKKK	KK	
RRRRRRRR	MM	MM	11	RRRRRRRR	EEEEEEEEEE	LL	BBBBBBBB	LL	KKKKKK	KK	
RR	RR	MM	11	RR	RR	LL	BB	LL	KK	KK	
RR	RR	MM	11	RR	RR	LL	BB	LL	KK	KK	
RR	RR	MM	11	RR	RR	LL	BB	LL	KK	KK	
RR	RR	MM	11	RR	RR	LL	BB	LL	KK	KK	
RR	RR	MM	111111	RR	RR	LLLLLLLLLL	BBBBBBBB	LLLLLLLLLL	KK	KK	
RR	RR	MM	111111	RR	RR	LLLLLLLLLL	BBBBBBBB	LLLLLLLLLL	KK	KK	

LL	IIIIII	SSSSSSSS
LL	IIIIII	SSSSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SSSSSS
LL	II	SSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LLLLLLLLLL	IIIIII	SSSSSSSS
LLLLLLLLLL	IIIIII	SSSSSSSS

(3) 84
(6) 159

DECLARATIONS
RMSRELBLK1 - ROUTINE TO RELEASE BLOCK, REWRITING IF DIRTY

```
0000 1 $BEGIN RM1RELBLK,000,RMSRMS1,<RELEASE BUFFER FOR SEQ. ORG.>
0000 2
0000 3 :
0000 4 :*****
0000 5 :*
0000 6 :* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 :* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 :* ALL RIGHTS RESERVED.
0000 9 :*
0000 10 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 :* TRANSFERRED.
0000 16 :*
0000 17 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 :* CORPORATION.
0000 20 :*
0000 21 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 :*
0000 24 :*
0000 25 :*****
0000 26 :
```



```
0000 28 :++
0000 29 : Facility: rms32
0000 30 :
0000 31 : Abstract:
0000 32 :         this module releases a buffer causing its
0000 33 :         contents to be written out if dirty.
0000 34 :
0000 35 : Environment:
0000 36 :         star processor running starlet exec.
0000 37 :
0000 38 : Author: L F Laverdure,      creation date: 29-march-77
0000 39 :
0000 40 : Modified By:
0000 41 :
0000 42 :         V03-005 DGB0018      Donald G. Blair      02-Mar-1984
0000 43 :         Use full-length fib to support access mode protected
0000 44 :         files.
0000 45 :
0000 46 :         V03-004 RAS0179      Ron Schaefer      29-Jul-1983
0000 47 :         Change default autoextend size to be 2*MBC now that
0000 48 :         we are defaulting MBC to a very large value.
0000 49 :         Remove use of volume default and clean-up retry logic.
0000 50 :
0000 51 :         V03-003 KBT0417      Keith B. Thompson    30-Nov-1982
0000 52 :         Change ifb$w_devbufsiz to ifb$l_devbufsiz
0000 53 :
0000 54 :         V03-002 RAS0095      Ron Schaefer      7-Sep-1982
0000 55 :         Correct auto-extend logic for overdraft. Repeal
0000 56 :         old algorithm that reported overdraft errors on the
0000 57 :         actually entry into overdraft as this lost allocated
0000 58 :         blocks.
0000 59 :
0000 60 :         V03-001 KBT0147      Keith B. Thompson    20-Aug-1982
0000 61 :         Reorganize psects
0000 62 :
0000 63 :         V02-020 JWH0002      Jeffrey W. Horn      12-Jan-1982
0000 64 :         Allow retry of auto-extend only on quota-exceeded and
0000 65 :         device-full errors from ACP.
0000 66 :
0000 67 :         V02-019 JWH0001      Jeffrey W. Horn      03-NOV-1981
0000 68 :         Changes to use new RMS default extend sizes.
0000 69 :
0000 70 :         V02-018 REFORMAT      P S Knibbe      25-Jul-1980
0000 71 :
0000 72 :         V017   PSK0009      P S Knibbe      10-JAN-1980      1:00
0000 73 :         changes to accomadate disk overdrafting. don't permit auto-extend
0000 74 :         if in overdraft. return error when first go into overdraft.
0000 75 :
0000 76 : Revision History
0000 77 :
0000 78 :         E H Marison, 7-JUN-1978 13:06
0000 79 :         add mbf logic and comments
0000 80 : 01 -
0000 81 : --
0000 82 :
```

```
0000 84      .SBTTL  DECLARATIONS
0000 85
0000 86 :
0000 87 : Include Files:
0000 88 :
0000 89 :
0000 90 :
0000 91 : Macros:
0000 92 :
0000 93 :
0000 94      $$$DEF
0000 95      $IFBDEF
0000 96      $IRBDEF
0000 97      $DEVDEF
0000 98      $BDBDEF
0000 99      $RLSDEF
0000 100     $FIBDEF
0000 101     $RMSDEF
0000 102
0000 103 :
0000 104 : Equated Symbols:
0000 105 :
0000 106 :
0000 107 :
0000 108 : Own Storage:
0000 109 :
0000 110
```



```
0000 112
0000 113 :++
0000 114 : notes on the multi-block buffering scheme
0000 115
0000 116 : this routine causes multiple blocks to be read together
0000 117 : (as specified by mbc) but returned one at a time for
0000 118 : processing by the calling routines.
0000 119
0000 120 : assumptions:
0000 121
0000 122 : 1. mbc is never > 0 except for disk(= # vbn's - 1)
0000 123 : 2. records are always written at eof (only updates
0000 124 : may occur elsewhere in the file).
0000 125 : 3. all sequential i/o calls go thru one of the
0000 126 : following routines:
0000 127 :     rm$xtblk1
0000 128 :     rm$wtlst1
0000 129 :     rm$relblk1
0000 130 : 4. there is no write sharing for sequential files.
0000 131 : 5. a direct release will be done only when there
0000 132 : is no i/o for the buffer.
0000 133
0000 134 : bdb field usage:
0000 135
0000 136 : 1. bdb$l_vbn = vbn of first block in buffer
0000 137 : 2. (irb$l_rp_vbn = vbn of current block)
0000 138 : 3. bdb$b_rel_vbn = current vbn rel to start vbn for buffer
0000 139 : 4. bdb$b_val_vbns = # of valid vbns in buffer
0000 140 : 5. bdb$b_flg$ :
0000 141 :     -bdb$b_drt: all blocks up to the greater of the current vbn
0000 142 :     and the number of val_vbns are dirty
0000 143 :     -bdb$b_val: the current vbn is valid
0000 144
0000 145 : 6. the relative vbn = requested vbn - start vbn
0000 146 : 7. current block buffer addr = buff addr + (rel_vbn*512)
0000 147 : 8. bdb$b_numb = # bytes in current block
0000 148 :     on reads = (irb$b_mbc+1)*512
0000 149 :     on writes = (max(val_vbn,rel_vbn+1))*512
0000 150 : 9. requested vbn is in buffer if its rel_vbn < or = mbc
0000 151 : 10. if read required and rel_vbn < val_vbns ok,
0000 152 :     else release buffer and reread
0000 153 : 11. on release (rm$relblk1) if bdb$b_val is off and the
0000 154 :     bdb$b_drt bit is set, merely decrement the
0000 155 :     current vbn and set the valid bit.
0000 156 :--
0000 157
```

```
0000 159      .SBTTL RM$RELBLK1 - ROUTINE TO RELEASE BLOCK, REWRITING IF DIRTY
0000 160
0000 161 :++
0000 162 : RM$RELBLK1 - Release block, rewrite if dirty
0000 163 :
0000 164 :   this routine releases the block whose bdb address is in r4.
0000 165 :   if the block is dirty it is first written.
0000 166 :   if the write is to a disk file, the file must have been extended if needed.
0000 167 :
0000 168 : Calling sequence:
0000 169 :
0000 170 :     bsbw    rm$relblk1
0000 171 :
0000 172 : Input Parameters:
0000 173 :
0000 174 :     r11     impure area address
0000 175 :     r10     ifab address
0000 176 :     r9      irab address
0000 177 :     r8      rab address
0000 178 :     r4      bdb address
0000 179 :
0000 180 : Implicit Inputs:
0000 181 :
0000 182 :     bdb$b_flg      -      (drt and val)
0000 183 :     bdb$b_rel_vbn  -      current block if mbc > 0
0000 184 :     bdb$w_num      -      # bytes to write if magtape
0000 185 :     bdb$l_vbn      -      start block for write
0000 186 :     bdb$l_addr     -      start address for write
0000 187 :     ifb$l_hbk      -      highest allocated block
0000 188 :     ifb$w_rtdeq    -      default extend quantity
0000 189 :     ifb$w_chnl     -      channel
0000 190 :     ifb$l_devbufsiz -      block buffer size
0000 191 :
0000 192 : Output Parameters:
0000 193 :
0000 194 :     r0      status code
0000 195 :     r1,ap   destroyed
0000 196 :
0000 197 : Implicit Outputs:
0000 198 :
0000 199 :     bdb$b_flg      -      drt is cleared
0000 200 :     -              -      val is set
0000 201 :     -              -      iop set if write behind underway
0000 202 :     bdb$w_num      -      set to size of transfer, if any
0000 203 :
0000 204 : Completion Codes:
0000 205 :
0000 206 :     standard rms, in particular, suc, sys, dme, wer, ful, prv, ext.
0000 207 :
0000 208 : Side Effects:
0000 209 :
0000 210 :     may have switched to running at ast level
0000 211 :     requiring reprobe of all user addresses not in rab.
0000 212 :--
0000 213
```



```
0000 215 RMSRELBLK1::
0000 216 $STSTPT RELBLK1
OC BB 0006 217 PUSHF #^M<R2,R3> ; save regs.
01 E1 0008 218 BBC #BDB$V_DRT,-
OA A4 000A 219 BDB$B_FLGS(R4),-
25 000C 220 RELEASE ; branch if buffer not dirty
1C E1 000D 221 #DEV$V_RND,-
6A 000F 222 IFB$S_PRIM_DEV(R10),-
21 0010 223 RELEASE ; branch if not disk
0011 224
0011 225 :
0011 226 : bdb marked dirty - adjust bdb data for release
0011 227 :
0011 228 :
0011 229 ASSUME BDB$V_VAL EQ 0
07 OA A4 E8 0011 230 BLBS BDB$B_FLGS(R4),10$ ; branch if buffer valid
0015 231
0015 232 :
0015 233 : buffer dirty but marked invalid.
0015 234 : decrement the current vbn data and say valid.
0015 235 :
0015 236 :
OA 48 A4 97 0015 237 DECB BDB$B_REL_VBN(R4)
OA A4 01 88 0018 238 BISB2 #BDB$M_VAL,BDB$B_FLGS(R4)
001C 239
001C 240 :
001C 241 : adjust byte count for transfer if disk
001C 242 : (i.e., transfer all blocks thru current vbn or # val_vbns if greater)
001C 243 :
001C 244 :
52 48 A4 9A 001C 245 10$: MOVZBL BDB$B_REL_VBN(R4),R2 ; get current vbn
52 B6 0020 246 INCW R2 ; get # vbns
49 A4 52 91 0022 247 CMPB R2,BDB$B_VAL_VBNS(R4) ; is current greater than # valid vbns?
04 1E 0026 248 BGEQU 15$ ; branch if yes
52 49 A4 9A 0028 249 MOVZBL BDB$B_VAL_VBNS(R4),R2 ; no - so use # valid vbns
52 A5 002C 250 15$: MULW3 R2,-
14 A4 48 AA 002E 251 IFB$S_DEVBUFFSIZ(R10),-
002E 252 BDB$W_NUMB(R4) ; and set xfer size
0032 253
0032 254 :
0032 255 : call rm$release to write and release buffer
0032 256 :
0032 257 :
53 02 D0 0032 258 RELEASE:
FFC8' 30 0032 259 MOVL #RL$M_WRT_THRU,R3 ; flag to get write
OC BA 0035 260 BSBW RMS$RELEASE ; release buffer
05 003A 261 POPR #^M<R2,R3>
RSB 262
```

```
003B 264 :++
003B 265 : RMS$AUTOEXTEND - Extend a sequential disk file
003B 266 :
003B 267 : this subroutine extends a sequential disk file.
003B 268 : the # of blocks for the extend is =
003B 269 : max (required # blocks, min (2*bdb$w_numb/512, 256)).
003B 270 : if this many blocks are not available only the minimum required are allocated.
003B 271 :
003B 272 : calling sequence:
003B 273 :
003B 274 :     bsbw    rm$autoextend
003B 275 :
003B 276 : input parameters:
003B 277 :
003B 278 :     r11      impure area address
003B 279 :     r10      ifab address
003B 280 :     r9       irab address
003B 281 :     r8       rab address
003B 282 :     r4       bdb address
003B 283 :     r2       minimum # of blocks to extend file
003B 284 :     bdb$w_numb number of bytes for xfer
003B 285 :
003B 286 : output parameters:
003B 287 :
003B 288 :     r0       status code
003B 289 :     r1,r3,ap destroyed
003B 290 :--
003B 291 :
003B 292 : RMS$AUTOEXTEND::
34  BB 003B 293 :     PUSH    #^M<R2,R4,R5>           ; save regs
003D 294 :
003D 295 :
003D 296 : build fib for extend
003D 297 :
003D 298 :
52  40 8F 9A 003D 299 :     MOVZBL  #FIB$C_LENGTH,R2       ; size of fib
51  5A  D0 0041 300 :     MOVL    R10,R1                 ; space header addr
    FFB9' 30 0044 301 :     BSBW    RM$GETSPC               ; allocate fib
    03 50  E8 0047 302 :     BLBS    R0,10$                 ; continue on success
    0081 31 004A 303 :     BRW     ERRXIT
18  A1  6E  D0 004D 304 10$:  MOVL    (SP),FIB$L_EXSZ(R1)       ; set # blocks required
    1A A1  B5 0051 305 :     TSTW    FIB$L_EXSZ+2(R1)       ; hi-order extend size zero?
    0B 12 0054 306 :     BNEQ    30$                    ; branch if not
4C  AA  6E  B1 0056 307 :     CMPW    (SP),IFB$W_RTDEQ(R10) ; extend size < default?
    05 1E 005A 308 :     BGEQU   30$                    ; branch if not
    4C AA  B0 005C 309 :     MOVW    IFB$W_RTDEQ(R10),-
    18 A1  005F 310 :     FIB$L_EXSZ(R1)                 ; yes - use default
    80 8F  90 0061 311 30$:  MOVB    #FIB$M_EXTEND,-
    16 A1  0064 312 :     FIB$W_EXCTL(R1)                 ; say it's an extend
    4C AA  B5 0066 313 :     TSTW    IFB$W_RTDEQ(R10)       ; zero default extend size?
    38 12 0069 314 :     BNEQ    DOXTND                 ; branch if not
    006B 315 :
    006B 316 :
    006B 317 : This is an auto extend with deq=0.
    006B 318 : Try to use system or process default extention quantity.
    006B 319 : Otherwise, compute the number of blocks to extend =
    006B 320 :     max(required,min(2*bufsiz,256))
```



```
006B 321 ;
006B 322 ;
50 00000000'9F 3C 006B 323 MOVZWL @#PIO$GW_RMSEXTEND,R0 ; Process deq?
22 12 0072 324 BNEQ 33$ ; branch if yes
50 00000000'9F 3C 0074 325 MOVZWL @#SYSS$GW_RMSEXTEND,R0 ; System deq?
19 12 007B 326 BNEQ 33$ ; branch if yes
50 14 A4 07 04 AE D0 007D 327 MOVL 4(SP),R4 ; restore bdb address
09 EF 0081 328 EXTZV #9,#7,BDB$W_NUMB(R4),R0 ; get # of blocks to xfer
02 A4 0087 329 MULW2 #2,R0 ; times 2
0100 8F 50 B1 008A 330 CMPW R0,#256 ; > max. default extend size?
05 15 008F 331 BLEQ 33$ ; branch if not
50 0100 8F B0 0091 332 MOVW #256,R0 ; yes - just extend 256 blocks
6E 50 D1 0096 333 33$: CMPL R0,(SP) ; > required extend size?
08 1B 0099 334 BLEQU 36$ ; branch if not
18 A1 50 B0 009B 335 MOVW R0,FIB$L_EXSZ(R1) ; use larger default extend size
009F 336 SSB #IFB$V_TEF,(R10) ; and say used so that we truncate
00A3 337 ; at eof on close
00A3 338 36$:
00A3 339
00A3 340 ;
00A3 341 ; build the fib descriptor
00A3 342 ;
00A3 343
7E 40 51 DD 00A3 344 DOXTND: PUSHL R1 ; addr of fib
8F 9A 00A5 345 MOVZBL #FIB$C_LENGTH,-(SP) ; length of fib
00A9 346
00A9 347 ;
00A9 348 ; do the extend
00A9 349 ;
00A9 350
53 FF54' 30 00A9 351 BSBW RM$FCPEXTEND ; & do extend
14 5A D0 00AC 352 MOVL R10,R3 ; get set for rm$retspc
03EC 8F 50 BA 00AF 353 POPR #^M<R2,R4> ; restore fib length (r2) and addr (r4)
37 13 00B1 354 CMPW R0,#SS$_EXDISKQUOTA ; hit overdraft ?
00B8 355 BEQLU OVERQUOTA ; spec handling if overquota
0850 8F 50 B1 00B8 356
15 13 00BD 357 CMPW R0,#SS$_DEVICEFULL ; if device full -
00BF 358 BEQLU RETRY ; retry
21 50 E9 00BF 359
00BF 360 BLBC R0,ERREXT ; if error code, exit
```



```
00C2 362
00C2 363 ;
00C2 364 ; extend complete.
00C2 365 ; update ifab hi block field, deallocate the fib, and return
00C2 366 ; return status can either be straight success, or SS$_OVRDSKQUOTA
00C2 367 ;
00C2 368
00C2 369          PUSH  R0          ; save success status
70 AA 1C A4 50 DD 00C2 370          ADDL3  FIB$_EXSZ(R4),FIB$_EX_VBN(R4),IFB$_HBK(R10)
18 A4 70 AA FF2F' 35 BA 00C4 371          DECL  IFB$_HBK(R10)          ; get # of highest allocated blk
00CB 372 ERRXIT: BSBW  RMS$RETSPC          ; return the fib space
00D1 373          POPR  #^M<R0,R2,R4,R5>          ; restore status and regs
00D3 374          RSB
00D4 375
00D4 376 ;
00D4 377 ; extend failed. if attempted extend was for more than minimum required amount,
00D4 378 ; retry extend for only the minimum required amount, otherwise
00D4 379 ; map error, release buffer & return
00D4 380 ;
00D4 381
18 A4 6E D1 00D4 382 RETRY: CMPL  (SP),FIB$_EXSZ(R4)          ;
00D8 383          BGEQU  ERREXT          ; branch if leq minimum
51 54 D0 00DA 384          MOVL  R4,R1          ; restore fib address
18 A1 6E D0 00DD 385          MOVL  (SP),FIB$_EXSZ(R1)          ; must extend this amount
C0 11 00E1 386          BRB  DOXTND          ; go retry extend
00E3 387
00E3 388 ;
00E3 389 ; errors
00E3 390 ;
00E3 391
00E3 392 ERREXT:
00E3 393          RMSERR  EXT,R1          ; default error code
FF15' 30 00E8 394          BSBW  RMS$MAPERR          ; map the error code
50 DD 00EB 395          PUSH  R0          ; save error code
DF 11 00ED 396          BRB  ERRXIT          ; cleanup and return
00EF 397
00EF 398 ;
00EF 399 ; we hit the over-quota mark on the disk.
00EF 400 ; if the user asked for this amount, then tell him.
00EF 401 ; if we guessed a value to extend by due to auto-extend, then tell him
00EF 402 ; but set the the amount to use from now on as 1 so as not to tie up blocks
00EF 403 ; now that we are close to the limit.
00EF 404 ;
00EF 405
00EF 406 OVERQUOTA:
4C AA B5 00EF 407          TSTW  IFB$_RTDEQ(R10)          ; if zero, then this is first time we're
00F2 408          ; in overdraft and we did auto-extend
EF 12 00F2 409          BNEQ  ERREXT          ; no - go away
4C AA B6 00F4 410          INCW  IFB$_RTDEQ(R10)          ; default extend is now 1, this disables
EA 11 00F7 411          ; auto-extend
00F7 412          BRB  ERREXT          ; return to mainline
00F9 413
00F9 414          .END
```


\$\$PSECT EP	=	00000000		
\$\$RMSTEST	=	0000001A		
\$\$RMS_PBUGCHK	=	00000010		
\$\$RMS_TBUGCHK	=	00000008		
\$\$RMS_UMODE	=	00000004		
BDB\$B_FLGS	=	0000000A		
BDB\$B_REL_VBN	=	00000048		
BDB\$B_VAL_VBNS	=	00000049		
BDB\$M_VAL	=	00000001		
BDB\$V_DRT	=	00000001		
BDB\$V_VAL	=	00000000		
BDB\$W_NUMB	=	00000014		
DEV\$V_RND	=	0000001C		
DOXTND		000000A3	R	01
ERREXT		000000E3	R	01
ERRXIT		000000CE	R	01
FIB\$C_LENGTH	=	00000040		
FIB\$C_EXSZ	=	00000018		
FIB\$C_EXVBN	=	000^001C		
FIB\$M_EXTEND	=	00000080		
FIB\$W_EXCTL	=	00000016		
IFB\$C_DEVBUFFSIZ	=	00000048		
IFB\$C_HBK	=	00000070		
IFB\$C_PRIM_DEV	=	00000000		
IFB\$V_TEF	=	00000036		
IFB\$W_RTDEQ	=	0000004C		
OVERQOTA		000000EF	R	01
PIOSA_TRACE		*****	X	01
PIOSGQ_RMSEXTEND		*****	X	01
RELEASE		00000032	R	01
RETRY		000000D4	R	01
RLSSM_WRT_THRU	=	00000002		
RMSAUTOEXTEND		0000003B	RG	01
RMSFCPEXTEND		*****	X	01
RMSGETSPC		*****	X	01
RMSMAPERR		*****	X	01
RMSRELBLK1		00000000	RG	01
RMSRELEASE		*****	X	01
RMSRETSPC		*****	X	01
RMS\$EXT	=	0001C022		
SS\$_DEVICEFULL	=	00000850		
SS\$_EXDISKQUOTA	=	000003EC		
SYSGW_RMSEXTEND		*****	X	01
TPT\$C_RELBLK1		*****	X	01

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes															
ABS	00000000 (0.)	00 (0.)	NOPI	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE					
RMSRMS1	000000F9 (249.)	01 (1.)	PIC	USR	CON	REL	GBL	NOSHR	EXE	RD	NOWRT	NOVEC	BYTE					
SAB\$	00000000 (0.)	02 (2.)	NOPI	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE					

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.10	00:00:01.07
Command processing	109	00:00:00.73	00:00:04.50
Pass 1	341	00:00:11.61	00:00:33.22
Symbol table sort	0	00:00:01.84	00:00:02.17
Pass 2	80	00:00:02.18	00:00:05.37
Symbol table output	7	00:00:00.09	00:00:00.12
Psect synopsis output	2	00:00:00.03	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	571	00:00:16.58	00:00:46.57

The working set limit was 1500 pages.

66017 bytes (129 pages) of virtual memory were used to buffer the intermediate code.

There were 70 pages of symbol table space allocated to hold 1321 non-local and 7 local symbols.

414 source lines were read in Pass 1, producing 13 object records in Pass 2.

22 pages of virtual memory were used to define 21 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
-\$255\$DUA28:[RMS.OBJ]RMS.MLB;1	10
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	0
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	7
TOTALS (all libraries)	17

1439 GETS were required to define 17 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:RM1RELBLK/OBJ=OBJ\$:RM1RELBLK MSRC\$:RM1RELBLK/UPDATE=(ENH\$:RM1RELBLK)+EXECML\$/LIB+LIB\$:RMS/LIB

0322

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY